UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

Natural Language Processing

Part-Of-Speech Tagging

LESSON 18

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Some of the slides are taken from DeepLearning.AI and from the slides accompanying the textbook by D. Jurafsky

HMM for POS Tagging

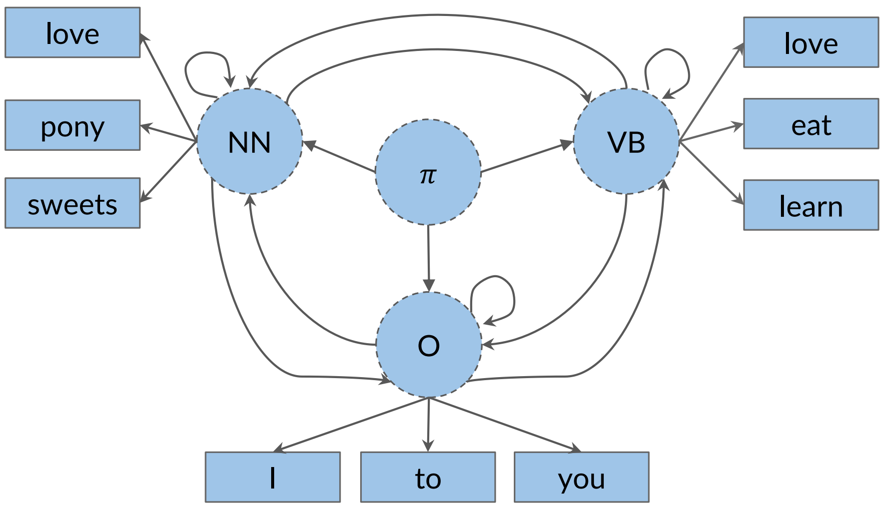
The Viterbi Algorithm

HMM tagging as decoding

- The task of determining the sequence of the hidden variables corresponding to the sequence of observations is called decoding
- Decoding
  - *Given as input an HMM with A and B matrices, and a sequence of observations $O=o_1,o_2, \ldots, o_T$, find the most probable sequence of states $Q=q_1q_2q_3\ldots q_T$*
  - For POS tagging, the goal of HMM decoding is to choose the tag sequence $t_1,\ldots,t_n$ that is most probable given the observation sequence of n words $w_1,\ldots,w_n$
- The decoding algorithm for HMMs is the Viterbi algorithm

Viterbi Algorithm: The big picture

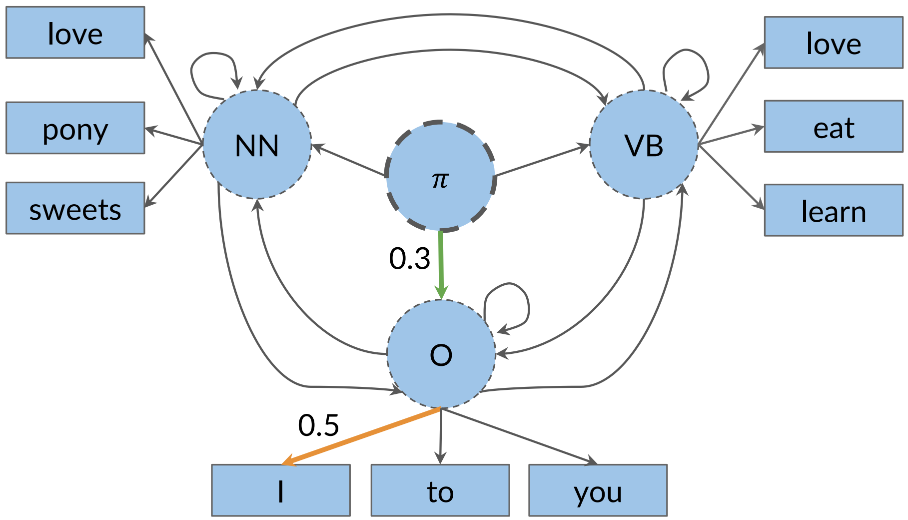
- Find the sequence of hidden states or parts of speech tags that have the highest probability for this sequence

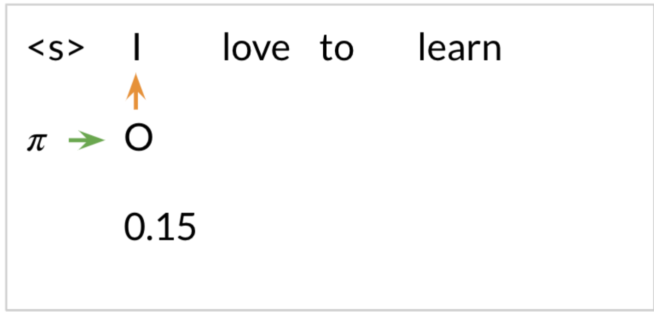


<s> I love to learn

Viterbi Algorithm: The big picture

- Let's start from the initial state π, selecting the next most probable hidden state

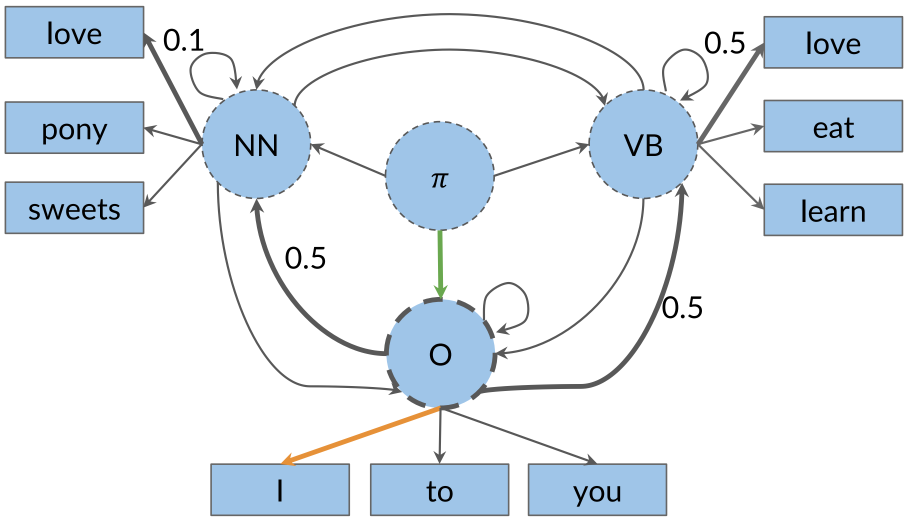


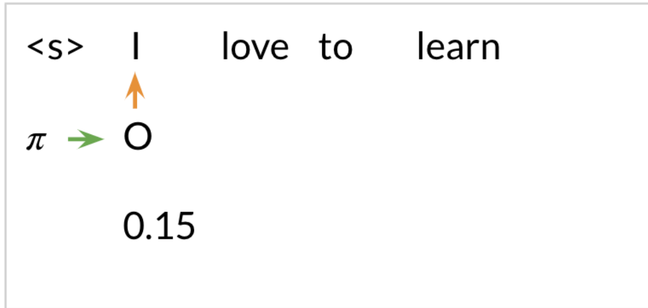


- The joint probability for observing the word *I* and with a transition through the O state is 0.15 (0.3 x 0.5, i.e., transition prob x emission prob)

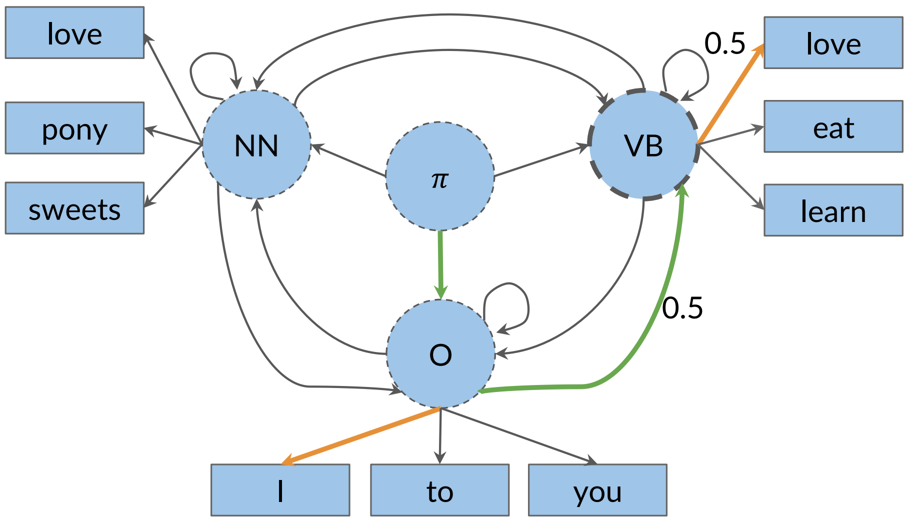
Viterbi Algorithm: The big picture

- Now, two possibilities of having observed the word *love*

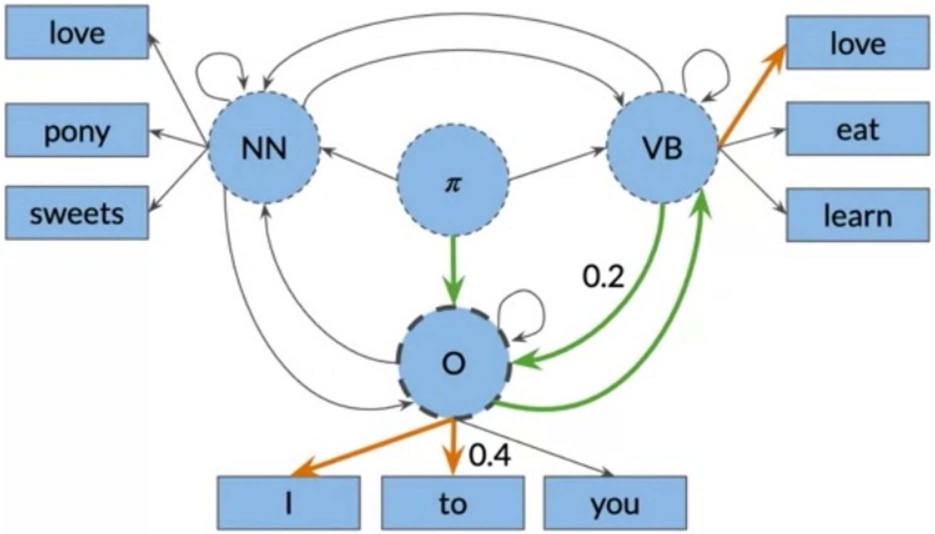




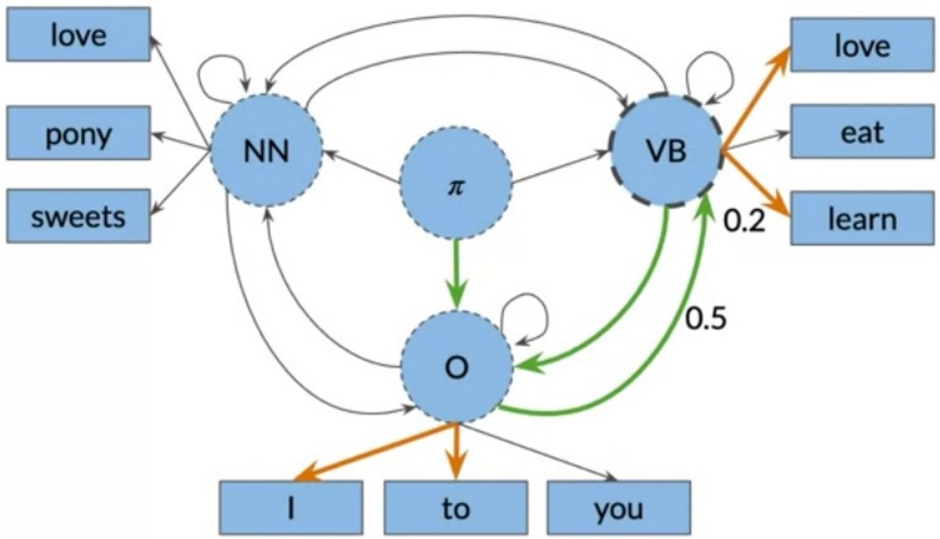
Viterbi Algorithm: The big picture

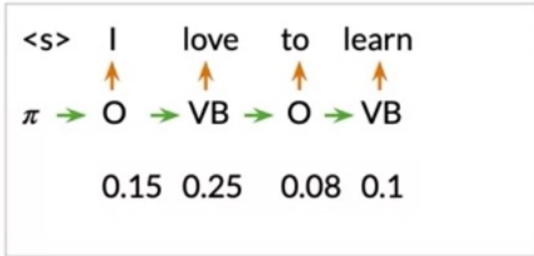


Viterbi Algorithm: The big picture



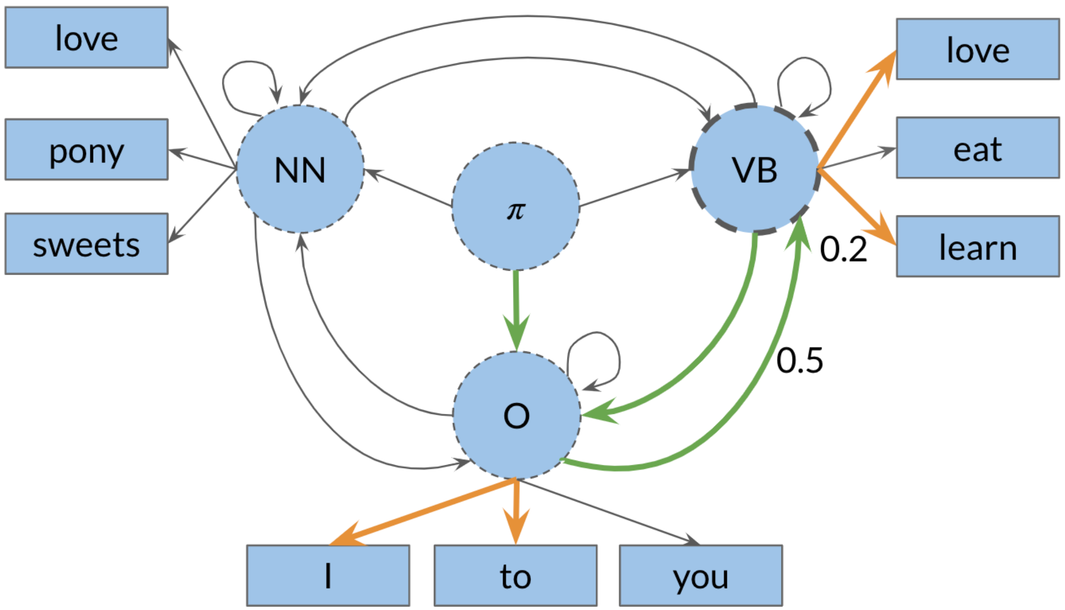
Viterbi Algorithm: The big picture

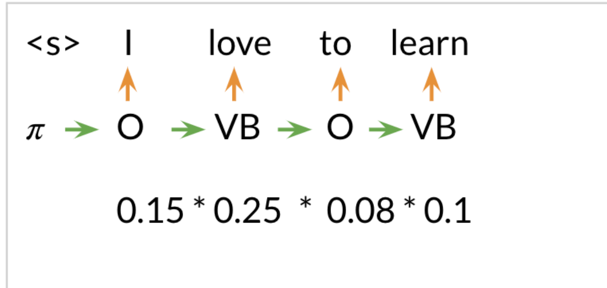




Viterbi Algorithm: The big picture

- The total probability is the product of all the probabilities for the single steps chosen





Probability for this sequence of hidden states: 0.0003

- The Viterbi algorithm computes several such paths at the same time for finding the most likely sequence of hidden states

Viterbi algorithm: Steps

- Initialization step
- Forward pass
- Backward pass

$C =$

	w_1	w_2	…	w_K
t_1				
…				
t_N				

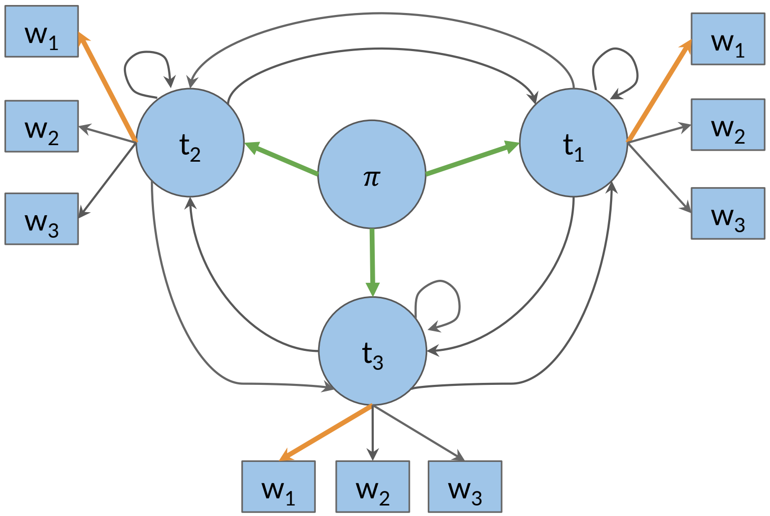
$D =$

	w_1	w_2	…	w_K
t_1				
…				
t_N				

- Auxiliary matrices
  - C holds the intermediate optimal probabilities
  - D holds the indices of the visited states
  - size NxK, N= number of POS tags, K = number of words in the given sequence

Initialization step: C matrix

- The first columns of matrices C and D are populated
  - C
    - The first column represents the probability of the transitions from the start state π to the first tag_i and the word w_1



$C =$

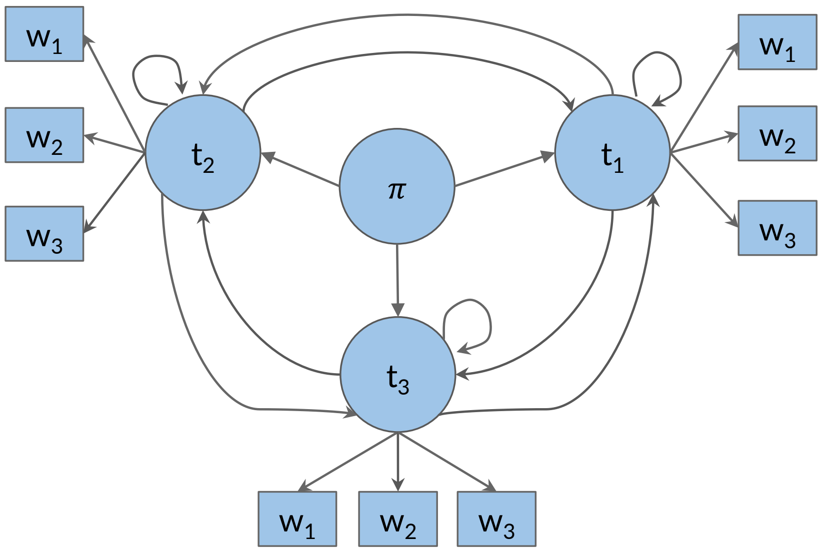
	w_1	w_2	...	w_K
t_1	$c_{1,1}$			
...				
t_N	$c_{N,1}$			

$$c_{i,1} = \pi_i * b_{i,cindex(w_1)}$$
$$= a_{1,i} * b_{i,cindex(w_1)}$$

  - *cindex(w_i)* returns the column index in the emission matrix B for a given word, w_i

Initialization step: D matrix

- D stores the labels that represent the different states we're traversing when finding the most likely sequence of POS tags from the given sequence of words, from w_1 to w_k
  - The first column has all 0 entries as there are no preceding POS tags traversed



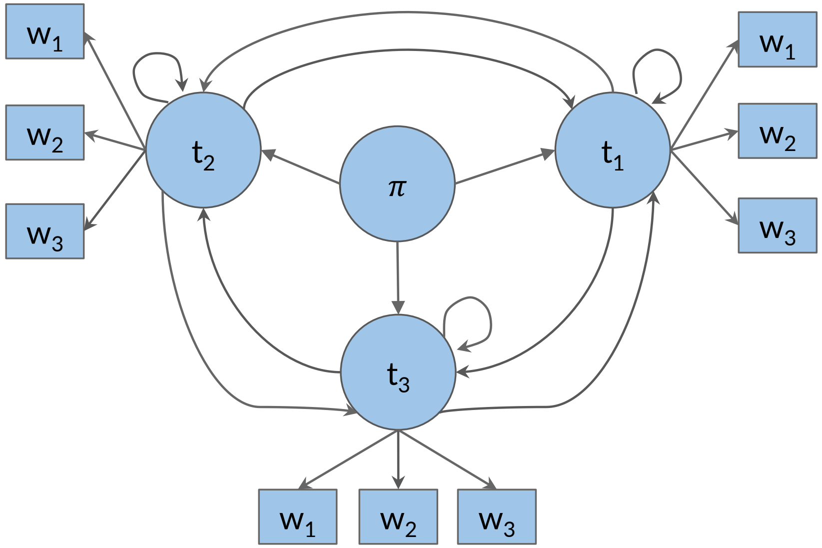
$D =$

	w_1	w_2	...	w_K
t_1	$d_{1,1}$			
...				
t_N	$d_{N,1}$			

$$d_{i,1} = 0$$

Forward pass

- C and D are populated column by column during the forward pass

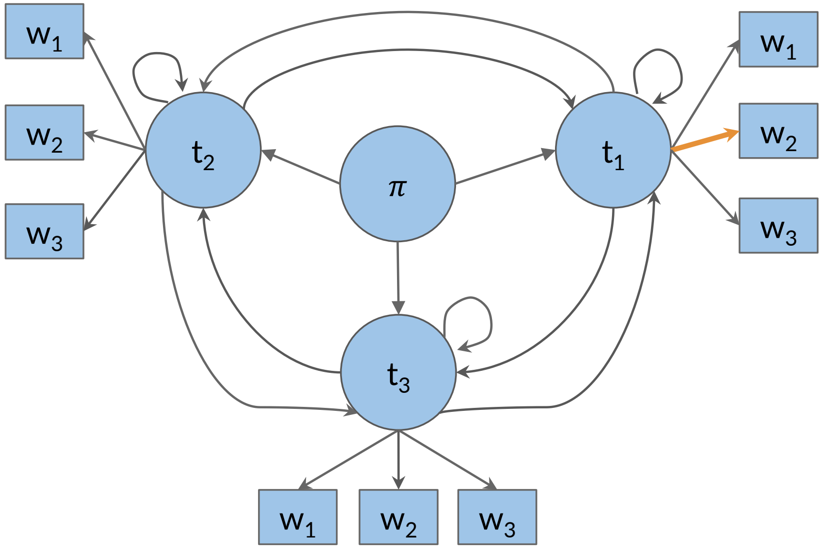


$C =$

	w_1	w_2	...	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
...				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$$c_{i,j} = \max_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

Forward pass

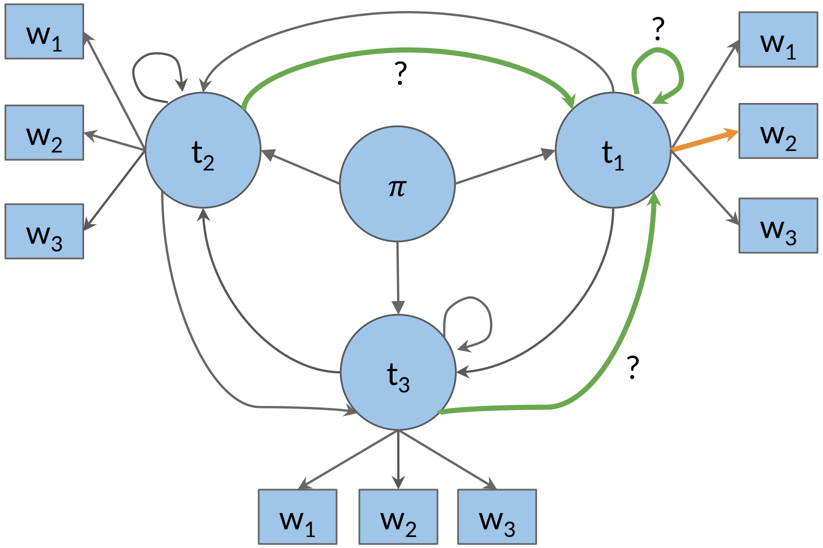


$C =$

	w1	w2	...	wK
t1	c1,1	c1,2		c1,K
...				
tN	cN,1	cN,2		cN,K

$$c_{1,2} = \max_k c_{k,1} * a_{k,1} * b_{1,cindex(w_2)}$$

Forward pass

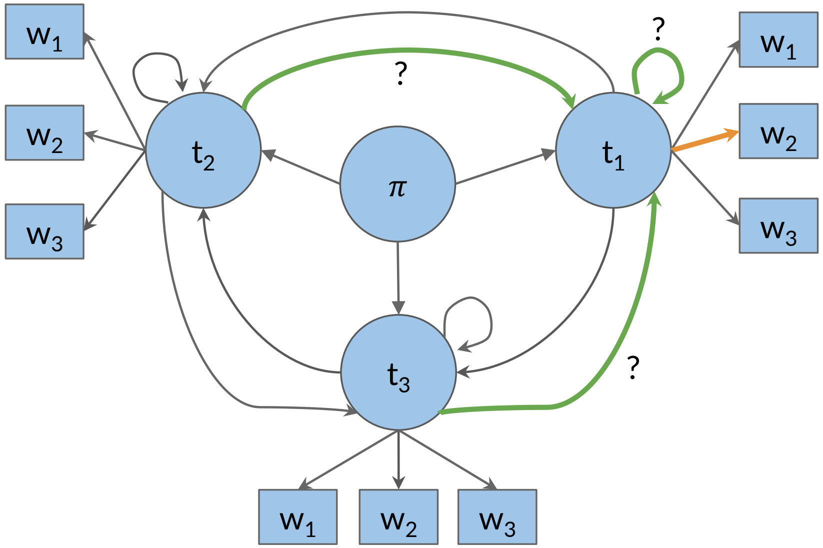


$C =$

	w_1	w_2	…	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
…				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$$c_{1,2} = \max_k c_{k,1} * a_{k,1} * b_{1,cindex(w_2)}$$

Forward pass

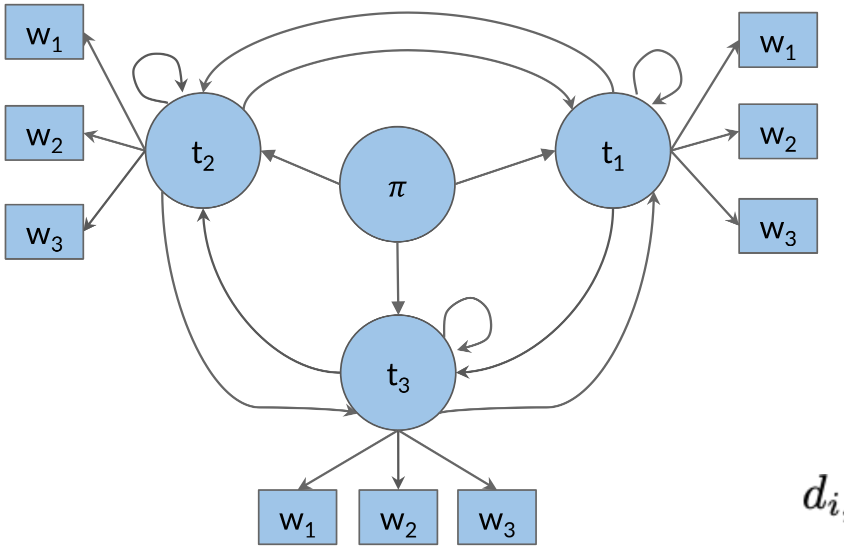


$$C =$$

	w_1	w_2	...	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
...				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$$c_{1,2} = \max_k c_{k,1} * a_{k,1} * b_{1,cindex(w_2)}$$

Forward pass



$D =$

	w_1	w_2	...	w_K
t_1	$d_{1,1}$	$d_{1,2}$		$d_{1,K}$
...				
t_N	$d_{N,1}$	$d_{N,2}$		$d_{N,K}$

$$c_{i,j} = \max_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

$$d_{i,j} = \operatorname{argmax}_k c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)}$$

- In each $d_{i,j}$ the k which maximizes the entry $c_{i,j}$ is stored

Backward pass

- The forward pass provided us the matrix C and D populated

$C =$

	w_1	w_2	...	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
...				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$D =$

	w_1	w_2	...	w_K
t_1	$d_{1,1}$	$d_{1,2}$		$d_{1,K}$
...				
t_N	$d_{N,1}$	$d_{N,2}$		$d_{N,K}$

$$s = \operatorname*{argmax}_i c_{i,K}$$

Backward pass

- First, calculate the index of the entry c_{iK} in the last column of C

$C =$

	w_1	w_2	…	w_K
t_1	$c_{1,1}$	$c_{1,2}$		$c_{1,K}$
…				
t_N	$c_{N,1}$	$c_{N,2}$		$c_{N,K}$

$D =$

	w_1	w_2	…	w_K
t_1	$d_{1,1}$	$d_{1,2}$		$d_{1,K}$
…				
t_N	$d_{N,1}$	$d_{N,2}$		$d_{N,K}$

$$s = \underset{i}{\operatorname{argmax}}\, c_{i,K}$$

Backward pass

- Example

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s> w1 w2 w3 w4 w5

Backward pass

$C =$

	w_1	w_2	w_3	w_4	w_5
t_1	0.25	0.125	0.025	0.0125	0.01
t_2	0.1	0.025	0.05	0.01	0.003
t_3	0.3	0.05	0.025	0.02	0.0000
t_4	0.2	0.1	0.000	0.0025	0.0003

$$s = \operatorname*{argmax}_i c_{i,K} = 1$$

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

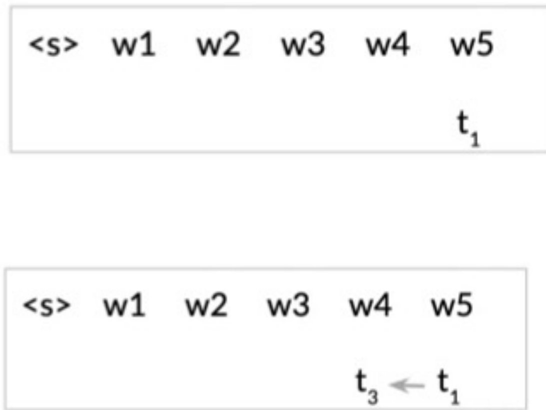
$$s = \underset{i}{\operatorname{argmax}}\, c_{i,K} = 1$$

<s> w1 w2 w3 w4 w5

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1



Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s> w1 w2 w3 w4 w5

$t_1 \leftarrow t_3 \leftarrow t_1$

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s> w1 w2 w3 w4 w5

$t_1 \leftarrow t_3 \leftarrow t_1$

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s> w1 w2 w3 w4 w5

$t_3 \leftarrow t_1 \leftarrow t_3 \leftarrow t_1$

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s> w1 w2 w3 w4 w5

$t_3 \leftarrow t_1 \leftarrow t_3 \leftarrow t_1$

Backward pass

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

<s>	w1	w2	w3	w4	w5
π	$\leftarrow t_2$	$\leftarrow t_3$	$\leftarrow t_1$	$\leftarrow t_3$	$\leftarrow t_1$

Named Entity Recognition

Named Entities

- In its core usage, a named entity means anything that can be referred to with a proper name
- Most common 4 tags:
  - PER (Person): "Marie Curie"
  - LOC (Location): "New York City"
  - ORG (Organization): "Stanford University"
  - GPE (Geo-Political Entity): "Boulder, Colorado"
  - Often multi-word phrases
    - But the term is also extended to things that aren't entities: dates, times, prices
- The task of named entity recognition (NER)
  - find spans of text that constitute proper names
  - tag the type of the entity

NER output

Citing high fuel prices, [$_{\text{ORG}}$ **United Airlines**] said [$_{\text{TIME}}$ **Friday**] it has increased fares by [$_{\text{MONEY}}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{\text{ORG}}$ **American Airlines**], a unit of [$_{\text{ORG}}$ **AMR Corp.**], immediately matched the move, spokesman [$_{\text{PER}}$ **Tim Wagner**] said. [$_{\text{ORG}}$ **United**], a unit of [$_{\text{ORG}}$ **UAL Corp.**], said the increase took effect [$_{\text{TIME}}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{\text{LOC}}$ **Chicago**] to [$_{\text{LOC}}$ **Dallas**] and [$_{\text{LOC}}$ **Denver**] to [$_{\text{LOC}}$ **San Francisco**].

Why NER?

- Sentiment analysis
  - consumer's sentiment toward a particular company or person?
- Question Answering
  - answer questions about an entity?
- Information Extraction
  - Extracting facts about entities from text

Why NER is hard

- Segmentation
  - In POS tagging, no segmentation problem since each word gets one tag
  - In NER we must find and segment the entities!
- Type ambiguity

> [$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
> [$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
> Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
> In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.

BIO Tagging

- How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

- [PER Jane Villanueva] of [ORG United], a unit of [ORG United Airlines Holding], said the fare applies to the [LOC Chicago] route

BIO Tagging

- [PER Jane Villanueva] of [ORG United], a unit of [ORG United Airlines Holding], said the fare applies to the [LOC Chicago] route

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

Now we have one tag per token!!!

BIO Tagging

- B: token that *begins* a span
- I: tokens *inside* a span
- O: tokens outside of any span

- # of tags (where n is #entity types):
  - 1 O tag,
  - *n* B tags,
  - *n* I tags
- total of *2n+1*

Words	**BIO Label**
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

BIO Tagging variants: IO and BIOES

- [PER Jane Villanueva] of [ORG United], a unit of [ORG United Airlines Holding], said the fare applies to the [LOC Chicago] route

Words	IO Label	BIO Label	BIOES Label
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Standard algorithms for NER

- Supervised Machine Learning given a human-labeled training set of text annotated with tags
  - Hidden Markov Models
  - Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
  - Neural sequence models (RNNs or Transformers)
  - Large Language Models (like BERT), finetuned